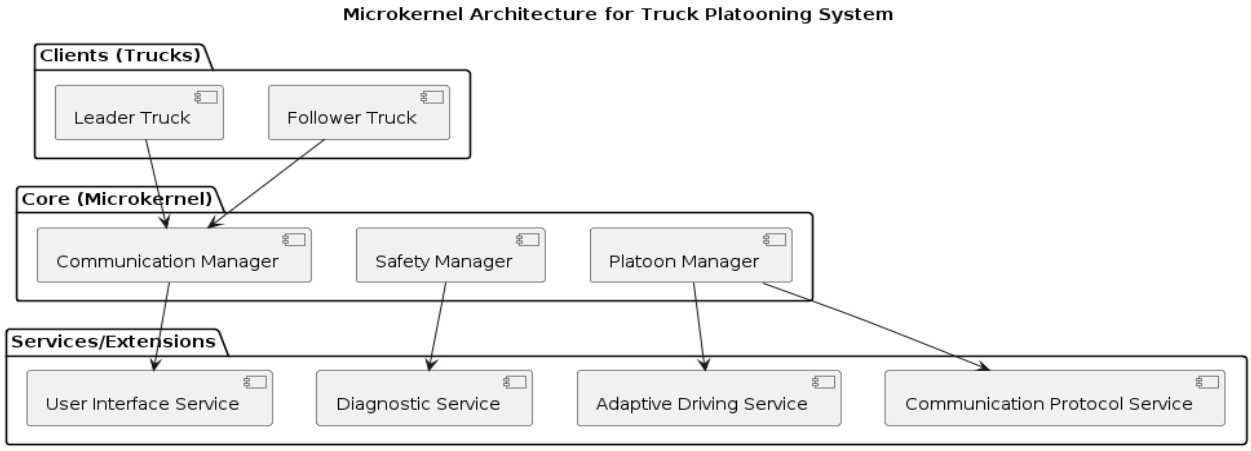
[Only for reference purpose]

Designing a Truck Platooning Systems

Message Queuing Pattern (from Concurrency Architectural design pattern) used in Microkernel pattern configuration.

Microkernel Architecture Pattern for Truck Platooning

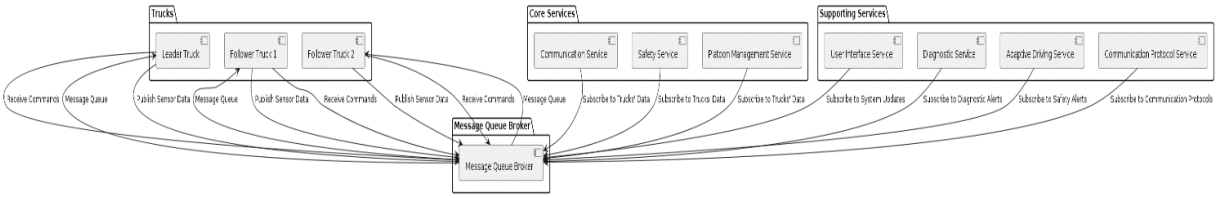


In designing a truck platooning system using the Message Queuing Pattern within a Microkernel Architecture, message queuing is essential for communication between the core microkernel and its external modules. The microkernel handles fundamental functions such as basic resource management and process communication. The primary packages include Clients (trucks), Core (Microkernel), and Services (extensions). Each truck (Client) interacts with the Core and Services through message queues, which allow for asynchronous and decoupled communication.

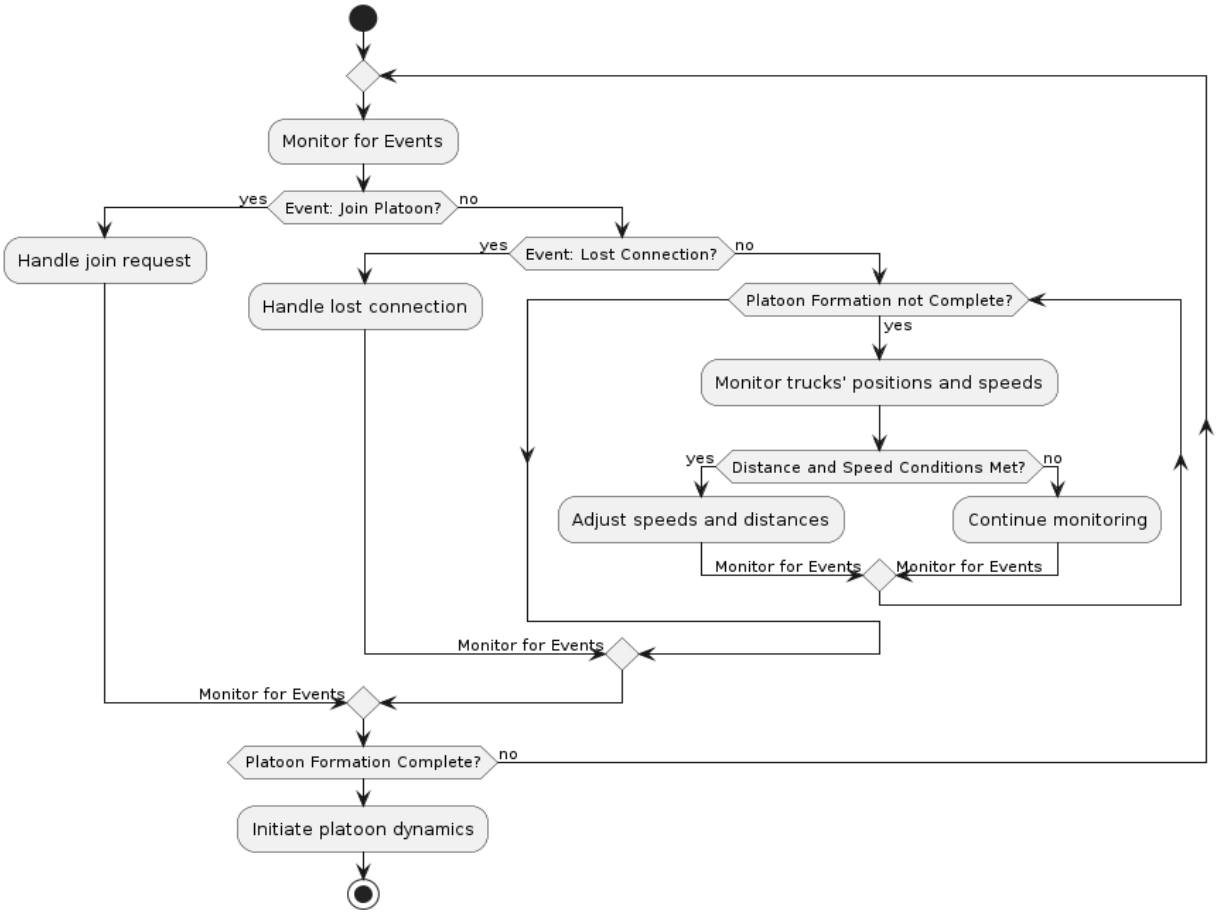
Within this system, message queues are implemented in the Inter-Process Communication (IPC) mechanisms of the microkernel and in the interface components of each service module. For example, a truck joining a platoon sends a join request to the Core via a message queue. The Core processes this request and communicates with other trucks through their respective queues. This setup ensures that messages are managed and prioritized efficiently, enabling the system to handle multiple tasks simultaneously and ensuring high-priority tasks are processed quickly. As a result, the system remains stable and responsive, even in the complex and dynamic environment of truck platooning.

Message Queuing

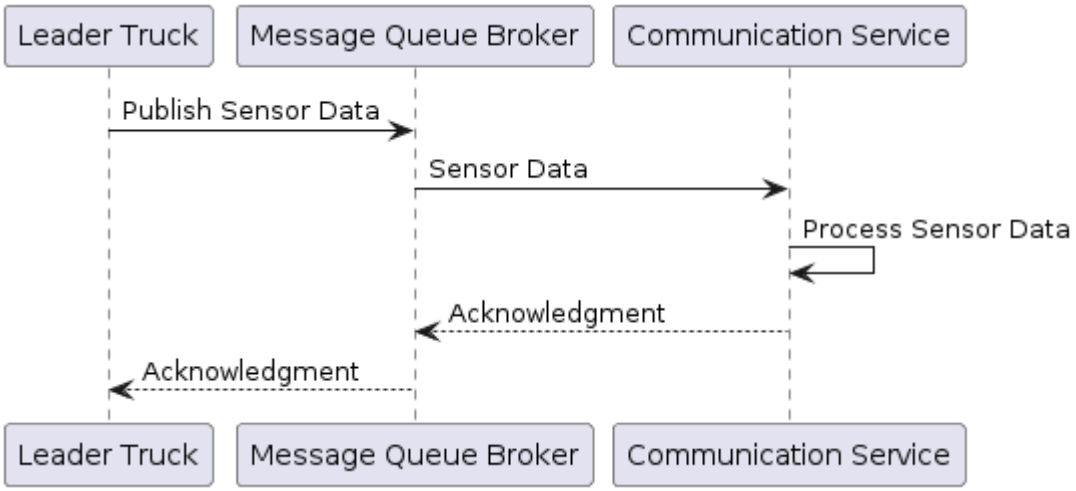
Components diagram



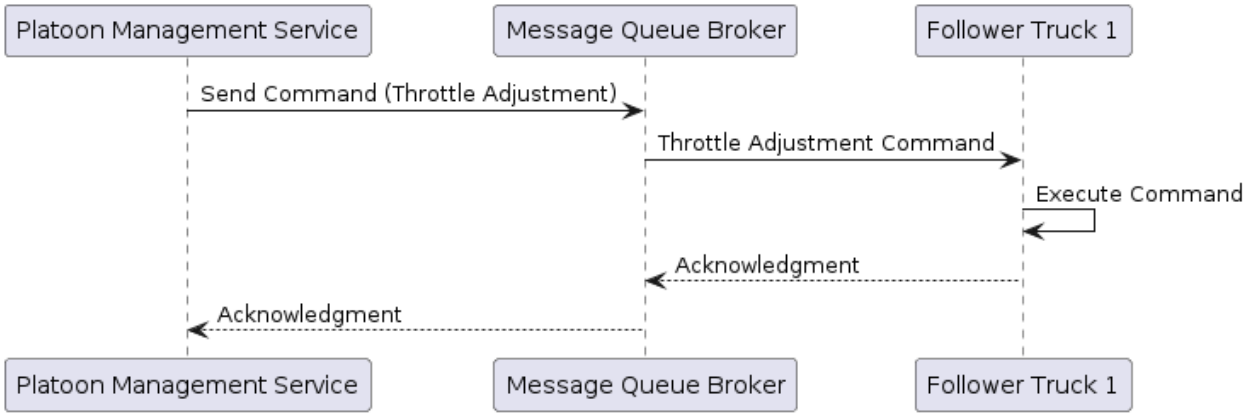
Activity Diagram



Sensor Data Publishing Sequence Diagram



Command Receiving Sequence Diagram



Reasons for choosing the Message Queuing Design Pattern

1. Why does the Message Queuing Pattern fit best?

The Message Queuing Pattern is well-suited for the truck platooning system due to several reasons:

Asynchronous Communication: Trucks in a platoon need to communicate with each other and with various services asynchronously. The Message Queuing Pattern allows for decoupled communication where trucks can publish sensor data (like GPS, radar, camera inputs) and receive control commands (such as throttle adjustments, braking commands) without waiting for immediate responses. This asynchronous nature supports real-time responsiveness and fault tolerance.

Scalability: In a platooning scenario, the number of trucks can vary, and each truck needs to communicate with multiple services (e.g., safety monitoring, platoon management, user interface). Message queues facilitate scalable communication by allowing multiple consumers (services) to subscribe to data from multiple producers (trucks) without overwhelming the system.

Reliability and Fault Tolerance: Message queues ensure reliable message delivery even if one of the trucks or services temporarily goes offline. Messages persist until they are successfully processed, reducing the risk of data loss and ensuring system reliability.

2. Why are other alternatives not as good?

Other architectural patterns may not be as suitable for a truck platooning system due to their inherent characteristics:

Client-Server Architecture: While this pattern could centralize some functionalities, it may introduce single points of failure and increase latency, which are critical issues in safety-critical systems like truck platooning.

Peer-to-Peer Architecture: This pattern lacks centralized control and may not provide robust mechanisms for coordinating platoon dynamics, safety monitoring, and adaptive driving algorithms effectively.

Layered Architecture: While it provides separation of concerns, it may not handle asynchronous communication and scalability requirements as efficiently as the Message Queuing Pattern, especially in dynamic and decentralized environments like truck platooning.

3. Mapping of Roles of the Architectural Design Pattern (see MQ component diagram)

Message Queue Broker: Acts as a central hub managing message queues. It facilitates communication between trucks (producers) and core/supporting services (consumers).

Trucks: Act as producers by publishing sensor data (e.g., GPS, radar, camera inputs) to message queues and consuming control commands (e.g., throttle adjustments, braking commands) from message queues.

Core Services:

Communication Service: Subscribes to message queues to receive and process data from trucks. Manages communication protocols and ensures reliable data exchange.

Safety Service: Monitors safety metrics (e.g., distance alerts, collision warnings) derived from sensor data received via message queues.

Platoon Management Service: Coordinates platoon formation, dissolution, and dynamics based on messages from trucks and safety service alerts.

Supporting Services:

User Interface Service: Subscribes to message queues to display real-time data and alerts to truck drivers and system operators.

Diagnostic Service: Monitors system health and performance metrics received via message queues, providing timely diagnostics and maintenance alerts.

Adaptive Driving Service: Utilizes sensor data and safety alerts from message queues to adjust driving behavior (e.g., adaptive cruise control, lane keeping) for optimal platooning performance.

Communication Protocol Service: Defines and manages communication protocols between trucks and services, ensuring interoperability and standardization.

4. Functionality of Each Component

Message Queue Broker: Manages creation, routing, and persistence of messages. Ensures reliable and efficient message delivery between trucks and services.

Trucks: Gather sensor data, publish to appropriate message queues, and respond to control commands received via message queues.

Core Services:

Communication Service: Handles message subscription, data processing, and ensures seamless communication between trucks and services.

Safety Service: Analyzes sensor data for safety-critical events, generates alerts, and communicates with platoon management service for corrective actions.

Platoon Management Service: Coordinates platoon operations, adjusts platoon configuration based on sensor data and safety alerts received via message queues.

Supporting Services:

User Interface Service: Subscribes to message queues to display real-time data, safety alerts, and system status updates to users.

Diagnostic Service: Monitors system health, receives diagnostic data via message queues, and alerts operators of any potential issues.

Adaptive Driving Service: Utilizes sensor data from message queues to adjust driving parameters for efficient and safe platooning operations.

Communication Protocol Service: Defines protocols, manages message formats, and ensures compatibility between diverse components in the system.

Extra diagram.

Components diagram with sensors and actuators